Electronic Version
Stylesheet Version v1.1.1

Claims

[c1] 1. A system for translation of data types between a first application in a first language and a second application in a second language, the system comprising:
a formal mapping between data types of the first language and data types of the second language;
translators for translating data types between the first language and the second language based on the formal mapping;
a translation mapping to the translators based on actual data types of the first application and formal data types of the second application; and
a module for selecting an appropriate translator for translating between a particular data type in the first language and a data type in the second language based on the translation mapping in response to invocation of a method of the first application with the particular data type.

[c2] 2. The system of claim 1, wherein the first language comprises C# and the second language comprises Java.

[c3] 3. The system of claim 1, wherein the formal mapping comprises a mapping between formal types of the first

language and formal types of the second language.

[c4] 4. The system of claim 3, wherein the formal types comprise static types.

[c5] 5. The system of claim 1, wherein the formal mapping comprises a many-to-one mapping.

[c6] 6. The system of claim 1, wherein the translators marshal translated data into a wire format for transfer from the first application to the second application across a network.

[c7] 7. The system of claim 1, wherein the translators read data of a first type and write data of a second type.

[c8] 8. The system of claim 1, wherein the translators include a mechanism for determining the actual type in the first language that a particular translator supports.

[c9] 9. The system of claim 1, wherein the translators include a mechanism for determining the formal type in the second language that a particular translator supports.

[c10] 10. The system of claim 1, wherein the translators provide information needed for creating the translation mapping.

[c11] 11. The system of claim 1, wherein the translators trans-

late return values received from the second application into a format appropriate for the first application.

[c12] 12. The system of claim 1, wherein the translation mapping provides for navigation from an object of the first application to a formal type of the second application's environment.

[c13] 13. The system of claim 1, wherein the translation mapping comprises a mapping from actual type of the first application and formal type of the second application to a particular translator.

[c14] 14. The system of claim 1, wherein the module for selecting an appropriate translator performs a two level lookup in the translation mapping.

[c15] 15. The system of claim 14, wherein the two level lookup includes a first level lookup based on actual data type of the first application.

[c16] 16. The system of claim 15, wherein the first level lookup considers inheritance hierarchy of the actual type.

[c17] 17. The system of claim 14, wherein the two level lookup includes a second level lookup based on formal data type of the second application.

[c18] 18. The system of claim 17, wherein the second level lookup selects the appropriate translator from a set of translators determined by the first level lookup.

[c19] 19. The system of claim 1, wherein the module for selecting an appropriate translator determines if the mapping includes at least one translator for the particular data type.

[c20] 20. The system of claim 1, wherein the module for selecting an appropriate translator determines if the mapping includes at least one translator for interfaces of the particular data type.

[c21] 21. The system of claim 1, wherein the module for selecting an appropriate translator determines if the mapping includes at least one translator for base types of the particular data type.

[c22] 22. A method for translation of data types between a first component in a first language and a second component in a second language, the method comprising: defining a formal mapping between data types of the first language and data types of the second language; implementing translators based on the formal mapping for translating data types between the first language and the second language;

producing a programming interface for the first component based upon the formal mapping and the second component's programming interface;
generating a translation mapping to the translators based on actual data types of the first component and formal data types of the second component as defined in the first component's programming interface;
in response to invocation of a method defined in the first component's programming interface with a particular data type, selecting a translator based on the translation mapping and the particular data type; and
translating the particular data type to a data type of the second language using the selected translator.

[c23] 23. The method of claim 22, wherein the first component comprises an application on a first machine and the second component comprises an application on a second machine.

[c24] 24. The method of claim 22, wherein the first component comprises a first component of an application and the second component comprises a second component of the application.

[c25] 25. The method of claim 22, wherein the first component and the second component operate within a single process.

[c26] 26. The method of claim 22, wherein the defining step includes defining a mapping between formal types of the first language and formal types of the second language.

[c27] 27. The method of claim 22, wherein the defining step includes defining a many-to-one mapping.

[c28] 28. The method of claim 22, wherein the implementing step includes implementing a translator for marshaling translated data into a wire format for transfer from the first component to the second component across a network.

[c29] 29. The method of claim 22, wherein the implementing step includes implementing a translator reading data of a first type and writing data of a second type.

[c30] 30. The method of claim 22, wherein the implementing step includes indicating the actual type in the first language that a particular translator supports.

[c31] 31. The method of claim 22, wherein the the implementing step includes indicating the formal type in the second language that a particular translator supports.

[c32] 32. The method of claim 22, wherein the generating step includes generating the translation mapping based, at least in part, on information provided by the translators.

[c33] 33. The method of claim 22, wherein the translation mapping provides for navigation from an object of the first component to the formal type of the second component's environment.

[c34] 34. The method of claim 22, wherein the translation mapping comprises a mapping from actual type of the first component and formal type of the second component to a particular translator.

[c35] 35. The method of claim 22, wherein the selecting step includes performing a two level lookup in the translation mapping.

[c36] 36. The method of claim 35, wherein the two level lookup includes a first level lookup based on actual data type of the first component.

[c37] 37. The method of claim 36, wherein the first level lookup considers inheritance hierarchy of the actual type.

[c38] 38. The method of claim 35, wherein the two level lookup includes a second level lookup based on formal data type of the second component.

[c39] 39. The method of claim 38, wherein the second level lookup includes selecting a translator from a set of

translators determined by the first level lookup based on formal data type.

[c40] 40. The method of claim 22, wherein the selecting step includes determining if the translation mapping includes at least one translator for the particular data type.

[c41] 41. The method of claim 22, wherein the selecting step includes determining if the translation mapping includes at least one translator for interfaces of the particular data type.

[c42] 42. The method of claim 22, wherein the selecting step includes determining if the translation mapping includes at least one translator for base types of the particular data type.

[c43] 43. The method of claim 22, further comprising: translating return values received from the second component into a data type of the first component's environment using the selected translator.

[c44] 44. The method of claim 22, wherein the first language is C# and the second language is Java.

[c45] 45. The method of claim 22, wherein the first language is Java and the second language is C#.

[c46] 46. A computer-readable medium having processor-

executable instructions for performing the method of claim 22.

[c47] 47. A downloadable set of processor-executable instructions for performing the method of claim 22.